PP4MNK Daedalus Main System Blueprint

A detailed technical report about my current EasyOS Daedalus setup, why I chose it as my main system, what I changed, how I changed it, and what each modification is meant to achieve

Author: pp4mnk
System: EasyOS Daedalus
Main philosophy: RAM-first computing, low SSD writes, portable browsers, clean sessions, high responsiveness, and practical daily stability.

---

1. Introduction

After testing different EasyOS and Puppy-based systems, I have finally reached a point where I am very satisfied with **EasyOS Daedalus**. At this stage, I believe Daedalus is going to become my **main daily operating system**.

This decision is not only based on appearance or personal preference. It comes after many practical tests involving:

OF COURSE, THERE COULD BE SOME TECHNICAL ERRORS, BUT THE MAIN IDEA IS THAT THIS EASYOS DAEDALUS MODDED WORKS FANTASTIC FOR ME... SO YOU CAN REPRODUCE IT... 4 SURE.

- boot behavior,
- RAM usage,
- browser performance,
- SSD write reduction,
- CPU governor behavior,
- streaming performance,
- JWM desktop customization,
- portable browser launchers,
- `/dev/shm` cache redirection,
- and general system responsiveness.

The result is a system that feels fast, clean, efficient, and very close to my ideal workflow: a lightweight desktop running mostly in RAM, with browsers optimized to reduce disk activity while keeping the user profile persistent.

In short, my goal has been to build a system that feels like this:

> **Fast like a live system, comfortable like an installed system, and careful with SSD writes like a RAM-centric workstation.**

---

2. Base system

My current setup is based on **EasyOS Daedalus**, running on older but still very capable hardware.

Typical hardware context:

```
CPU: Intel Core i5 680 @ 3.60 GHz
Architecture: x86_64
RAM: around 11 GiB
Graphics: Intel i915 / Ironlake generation
Desktop: JWM
Terminal: Sakura
Storage: SSD
```

This is not modern high-end hardware, but with the right operating system and careful configuration, it can still feel extremely responsive.

That is one of the main reasons why EasyOS Daedalus is such a good fit: it is lightweight enough to respect older hardware, but flexible enough to allow advanced RAM-based workflows.

---

3. Why I chose Daedalus as my main system

I tested other systems and configurations, including Excalibur and XFCE/Fusilli-style setups. Some of them are excellent, and some may even feel more visually advanced or feature-rich.

However, Daedalus currently gives me the best balance between:

- stability,
- simplicity,
- speed,
- low resource usage,
- predictable boot behavior,
- JWM compatibility,
- portable browser workflow,
- RAM-based optimizations,
- and manual control.

Daedalus feels like a very good base for a **personal performance workstation**.

It does not try to do too much. It gives me enough control to build my own environment around it.

For my use case, that is exactly what I want.

---

4. Main design philosophy

The system is built around five main ideas.

4.1 Run as much as possible in RAM

RAM is much faster than SSD storage. When possible, temporary data, browser caches, GPU caches and temporary files should be redirected to RAM.

This improves responsiveness and reduces unnecessary disk writes.

4.2 Keep profiles persistent, but caches temporary

I do not want to lose my browser profile every time I reboot. Bookmarks, sessions, extensions, preferences and logins are useful.

However, I do not need browser cache files to remain permanently stored on disk.

So the goal is:

```
Persistent browser profile: YES
Persistent browser cache: NO
Cache in RAM: YES
```

This gives the best compromise between usability and disk protection.

4.3 Reduce SSD writes

One of my main goals is to reduce unnecessary writes to the SSD.

This is useful for several reasons:

- it reduces wear on the SSD,
- it keeps the system cleaner,
- it avoids uncontrolled browser write activity,
- it makes the system more predictable,
- and it supports the EasyOS philosophy of controlled saving.

4.4 Keep the desktop lightweight

I currently use **JWM**, not Plank and not a heavy desktop environment.

This keeps the system simple and fast.

The desktop can still look modern and elegant with the right theme, icons, tray layout, dark bars and a clean wallpaper.

4.5 Prefer practical performance over theoretical perfection

Not every flag or tweak improves real performance. Some browser flags look impressive but can make older Intel hardware less stable or consume more CPU.

My approach is practical:

- test,
- observe,
- compare,
- keep what improves real use,
- remove what causes instability or unnecessary writes.

---

5. Boot strategy: RAM-oriented Daedalus

One of the most important parts of my setup is the boot configuration.

I use a RAM-oriented boot mode with parameters such as:

```
qfix=copy,nosave
```

A typical kernel command line looks like this:

```
rw wkg_uuid=xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx wkg_dir=pp4mnkos
qfix=copy,nosave ramdisk_size=4096000 loglevel=3 nowatchdog
```

What this means

`qfix=copy`

This tells EasyOS to copy important system components into RAM where possible.

The purpose is to make the session faster and reduce dependency on constant disk access.

`nosave`

This prevents automatic saving of the session.

The purpose is to avoid uncontrolled writes and keep the system clean unless I explicitly decide to save something.

`ramdisk_size=4096000`

This gives enough RAM space for the RAM-based working environment.

`loglevel=3`

This reduces kernel verbosity.

It keeps boot messages cleaner and avoids unnecessary noise.

`nowatchdog`

This disables the watchdog.

On some older systems, this may reduce unnecessary background activity or avoid certain watchdog-related delays.

---

6. Verifying RAM mode

I do not only assume that the system is running in a RAM-friendly mode. I verify it.

Useful commands include:

```
cat /proc/cmdline
```

```
cat /etc/rc.d/PUPSTATE
```

```
mount | grep -E "tmpfs|overlay|aufs|union|easy_ro|easy_rw|pup_ro|pup_rw"
```

```
df -h | grep -E "tmpfs|overlay|/initrd|easy_|pup_"
```

These commands show whether the system is using RAM layers, overlays, tmpfs, or zram-related structures.

The goal is to confirm that the system is not constantly depending on SSD writes for temporary runtime activity.

---

7. Monitoring write activity

To check whether the system is writing heavily or not, I use commands such as:

```
grep Dirty /proc/meminfo
```

Example result:

```
Dirty:              1020 kB
```

A low `Dirty` value is a very good sign.

It means there is not a huge amount of unwritten data waiting to be flushed to disk.

This is especially important when testing browser cache redirection.

If browsers are correctly using `/dev/shm`, SSD write pressure should be much lower.

---

8. Browser strategy

Browsers are usually among the most aggressive applications when it comes to disk writes.

They write:

- cache files,
- GPU cache,
- media cache,
- temporary files,
- session data,
- extension data,
- crash reports,
- history databases,
- cookies,
- profile metadata.

The goal is not to make browsers completely stateless. That would be uncomfortable.

The goal is to separate what must be persistent from what can safely be temporary.

Persistent

I want to keep:

- bookmarks,
- logins,
- browser preferences,
- extensions,
- useful profile state.

Temporary in RAM

I want to move to RAM:

- disk cache,
- GPU cache,
- media cache,
- temporary files,
- browser temp directories.

This is done using `/dev/shm`.

---

9. Why `/dev/shm`

`/dev/shm` is a tmpfs filesystem stored in RAM.

It is ideal for browser caches because:

- it is very fast,
- it is automatically cleared after reboot,
- it avoids SSD writes,
- it improves temporary file performance,
- it allows clean browser cache sessions.

This fits perfectly with a RAM-centric EasyOS setup.

My browser RAM cache structure is usually like this:

```
/dev/shm/pp4mnk-browsers/
├── opera/
│   ├── cache
│   ├── gpu-cache
│   ├── media-cache
│   └── tmp
├── slimjet/
│   ├── cache
│   ├── gpu-cache
│   └── tmp
├── chromium/
│   ├── cache
│   ├── gpu-cache
│   ├── media-cache
│   └── tmp
└── firefox/
    ├── cache
    └── tmp
```

This gives each browser its own separate RAM working area.

---

10. Opera RAM launcher

Opera was one of the browsers I wanted to optimize because it can write a lot of data during normal browsing and streaming.

The goal of this launcher is:

- keep the Opera profile persistent,
- move cache to RAM,
- move GPU cache to RAM,

- move media cache to RAM,
- move temporary files to RAM,
- reduce SSD writes,
- keep the browser practical for daily use.

Script: PP4MNK Opera RAM Launcher

```
#!/bin/sh
# ========================================
# PP4MNK OPERA RAM LAUNCHER
# Persistent profile + cache in RAM
# Goal: reduce SSD writes without losing browser profile
# ========================================

RAM_BASE="/dev/shm/pp4mnk-browsers/opera"

CACHE_DIR="$RAM_BASE/cache"
GPU_CACHE_DIR="$RAM_BASE/gpu-cache"
MEDIA_CACHE_DIR="$RAM_BASE/media-cache"
TMP_DIR="$RAM_BASE/tmp"

mkdir -p "$CACHE_DIR" "$GPU_CACHE_DIR" "$MEDIA_CACHE_DIR" "$TMP_DIR"

export TMPDIR="$TMP_DIR"
export XDG_CACHE_HOME="$CACHE_DIR"

exec /files/portable/Opera-portable64/RUN
 --disk-cache-dir="$CACHE_DIR"
 --gpu-disk-cache-dir="$GPU_CACHE_DIR"
 --media-cache-dir="$MEDIA_CACHE_DIR"
 --disk-cache-size=536870912
 --media-cache-size=268435456
 --no-first-run
 --no-default-browser-check
 "$@"
```

Explanation

`RAM_BASE="/dev/shm/pp4mnk-browsers/opera"`

This defines Opera's temporary RAM workspace.

Everything inside this directory is temporary and stored in RAM.

`CACHE_DIR`

This is where Opera stores its disk cache.

Despite the name “disk cache”, it is redirected to RAM.

`GPU_CACHE_DIR`

This redirects GPU-related browser cache to RAM.

This is useful because Chromium-based browsers can write GPU cache files frequently.

`MEDIA_CACHE_DIR`

This redirects media cache to RAM.

It is especially useful for streaming and video-heavy browsing.

`TMP_DIR`

This gives Opera a RAM-based temporary directory.

`export TMPDIR="$TMP_DIR"`

This tells applications launched from the script to use this directory for temporary files.

`export XDG_CACHE_HOME="$CACHE_DIR"`

This redirects generic XDG cache activity to the RAM cache directory.

`--disk-cache-dir`

This forces the browser disk cache to use RAM.

`--gpu-disk-cache-dir`

This forces GPU cache to use RAM.

`--media-cache-dir`

This forces media cache to use RAM.

`--disk-cache-size=536870912`

This sets browser disk cache size to 512 MiB.

It is large enough for normal browsing but not excessive.

`--media-cache-size=268435456`

This sets media cache size to 256 MiB.

It is useful for streaming without wasting too much RAM.

`"$@"`

This passes any URL or extra argument from the JWM tray or terminal to Opera.

Example:

```
/files/portable/opera-ram.sh https://chatgpt.com
```

---

11. Slimjet RAM launcher

Slimjet is one of my preferred browsers for streaming and daily browsing because it feels fast and practical.

The goal is similar:

- keep the profile,
- move cache to RAM,
- reduce writes,
- improve responsiveness,
- avoid excessive flags that could hurt stability on old Intel graphics.

Script: PP4MNK Slimjet RAM Launcher

```
#!/bin/sh
# ========================================
# PP4MNK SLIMJET RAM LAUNCHER
# Portable Slimjet + cache in RAM
# Optimized for responsiveness and reduced SSD writes
# ========================================

RAM_BASE="/dev/shm/pp4mnk-browsers/slimjet"

CACHE_DIR="$RAM_BASE/cache"
GPU_CACHE_DIR="$RAM_BASE/gpu-cache"
TMP_DIR="$RAM_BASE/tmp"

mkdir -p "$CACHE_DIR" "$GPU_CACHE_DIR" "$TMP_DIR"

export TMPDIR="$TMP_DIR"
export XDG_CACHE_HOME="$CACHE_DIR"

exec /files/portable/Slimjet-portable64_streaming_v49.0.2/Slimjet-portable64/
RUN-turbo-STREAMING
 --disk-cache-dir="$CACHE_DIR"
 --gpu-disk-cache-dir="$GPU_CACHE_DIR"
 --disk-cache-size=536870912
 --media-cache-size=268435456
 --disable-session-crashed-bubble
 --no-first-run
 --password-store=basic
 "$@"
```

Explanation

This script keeps Slimjet's portable profile in its original location while moving volatile cache data to RAM.

The most important options are:

```
--disk-cache-dir
--gpu-disk-cache-dir
--disk-cache-size
--media-cache-size
```

These reduce SSD write pressure while preserving usability.

The option:

```
--disable-session-crashed-bubble
```

is useful because in RAM/nosave workflows the browser may sometimes think it was closed incorrectly. This flag reduces the annoying restore-session prompt.

The option:

```
--password-store=basic
```

can be useful in portable/root-style environments where desktop keyrings may not behave like in a standard mainstream Linux installation.

---

12. Chromium RAM launcher

Chromium is useful as a clean, general-purpose browser and for testing.

The same cache-in-RAM logic applies.

Script: PP4MNK Chromium RAM Launcher

```
#!/bin/sh
# ========================================
# PP4MNK CHROMIUM RAM LAUNCHER
# Chromium cache, GPU cache, media cache and TMP in RAM
# ========================================

RAM_BASE="/dev/shm/pp4mnk-browsers/chromium"

CACHE_DIR="$RAM_BASE/cache"
```

```
GPU_CACHE_DIR="$RAM_BASE/gpu-cache"
MEDIA_CACHE_DIR="$RAM_BASE/media-cache"
TMP_DIR="$RAM_BASE/tmp"

mkdir -p "$CACHE_DIR" "$GPU_CACHE_DIR" "$MEDIA_CACHE_DIR" "$TMP_DIR"

export TMPDIR="$TMP_DIR"
export XDG_CACHE_HOME="$CACHE_DIR"

exec chromium
 --disk-cache-dir="$CACHE_DIR"
 --gpu-disk-cache-dir="$GPU_CACHE_DIR"
 --media-cache-dir="$MEDIA_CACHE_DIR"
 --disk-cache-size=536870912
 --media-cache-size=268435456
 --disable-session-crashed-bubble
 --no-first-run
 --no-default-browser-check
 "$@"
```

Verification

After launching Chromium, I can check RAM cache usage with:

```
du -sh /dev/shm/pp4mnk-browsers/chromium/*
```

Example output:

```
0    /dev/shm/pp4mnk-browsers/chromium/cache
0    /dev/shm/pp4mnk-browsers/chromium/gpu-cache
0    /dev/shm/pp4mnk-browsers/chromium/media-cache
0    /dev/shm/pp4mnk-browsers/chromium/tmp
```

And write pressure with:

```
grep Dirty /proc/meminfo
```

If `Dirty` remains very low, the setup is working well.

---

13. Firefox RAM approach

Firefox does not use exactly the same Chromium flags, but the goal is still the same:

- keep the profile persistent,
- move temporary and cache activity to RAM,

- reduce disk writes,
- keep browsing fast and clean.

A practical Firefox launcher can use RAM variables and profile preferences.

Script: PP4MNK Firefox RAM Launcher

```
#!/bin/sh
# ========================================
# PP4MNK FIREFOX RAM LAUNCHER
# Persistent profile + TMP/cache in RAM
# ========================================

RAM_BASE="/dev/shm/pp4mnk-browsers/firefox"
CACHE_DIR="$RAM_BASE/cache"
TMP_DIR="$RAM_BASE/tmp"

mkdir -p "$CACHE_DIR" "$TMP_DIR"

export TMPDIR="$TMP_DIR"
export XDG_CACHE_HOME="$CACHE_DIR"

exec /files/portable/Firefox-portable64/firefox/firefox
 --profile /files/portable/Firefox-portable64/profile
 "$@"
```

Recommended Firefox profile preferences

Inside Firefox, some cache behavior can also be controlled from `about:config`.

Useful preferences include:

```
browser.cache.disk.enable = false
browser.cache.memory.enable = true
browser.cache.memory.capacity = -1
```

Explanation

`browser.cache.disk.enable = false`

Disables persistent disk cache.

`browser.cache.memory.enable = true`

Enables memory cache.

`browser.cache.memory.capacity = -1`

Allows Firefox to manage memory cache automatically.

This fits well with an 11 GiB RAM system.

---

14. JWM tray launchers

Since I use JWM, I launch many browsers and web apps from tray buttons.

A simple launcher line looks like this:

```
<TrayButton popup="ChatGPT" icon="/usr/share/icons/blue_chatgpt.png">exec:/
files/portable/firefox-ram.sh https://chatgpt.com</TrayButton>
```

Why simple launcher lines are better

For Firefox, this is better:

```
<TrayButton popup="ChatGPT" icon="/usr/share/icons/blue_chatgpt.png">exec:/
files/portable/firefox-ram.sh https://chatgpt.com</TrayButton>
```

Than this:

```
<TrayButton popup="ChatGPT" icon="/usr/share/icons/blue_chatgpt.png">exec:/
files/portable/firefox-ram.sh --disk-cache-size=8048576000 --media-cache-
size=8048576000 https://chatgpt.com</TrayButton>
```

Why?

Because Firefox does not use Chromium-style `--disk-cache-size` and `--media-cache-size` flags in the same way.

Adding huge cache flags to Firefox does not improve performance and may create confusion.

The clean approach is:

```
Firefox cache behavior: controlled by Firefox preferences and RAM environment
Chromium-based cache behavior: controlled by launch flags
```

---

15. Opera / Slimjet / Chromium flag philosophy

With Chromium-based browsers, it is tempting to add many flags.

However, more flags do not always mean better performance.

On older Intel i915 / Ironlake graphics, aggressive GPU flags can sometimes create instability, higher CPU usage, or worse video behavior.

My safe philosophy is:

Keep

```
--disk-cache-dir
--gpu-disk-cache-dir
--media-cache-dir
--disk-cache-size
--media-cache-size
--disable-session-crashed-bubble
--no-first-run
--no-default-browser-check
```

Use carefully

```
--enable-gpu-rasterization
--enable-zero-copy
--ignore-gpu-blocklist
--enable-accelerated-video-decode
```

These may help in some cases, but they must be tested.

Avoid excessive or repeated flags

For example, repeating this is unnecessary:

```
--disable-infobars --disable-infobars
```

One copy is enough.

---

16. CPU governor optimization

Another important part of the setup is CPU behavior.

I prefer the `ondemand` governor because it gives a good balance between performance and efficiency.

The goal is:

- keep the system cool when idle,
- increase CPU frequency quickly when needed,
- avoid permanent maximum frequency,
- keep the desktop responsive.

Check current governor

```
cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

Set ondemand governor

```
for CPU in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor; do
  echo ondemand > "$CPU"
done
```

Tune ondemand behavior

```
echo 95 > /sys/devices/system/cpu/cpufreq/ondemand/up_threshold
echo 10 > /sys/devices/system/cpu/cpufreq/ondemand/sampling_down_factor
echo 200 > /sys/devices/system/cpu/cpufreq/ondemand/powersave_bias
```

Explanation

`up_threshold=95`

The CPU increases frequency when load is high enough.

This avoids unnecessary frequency jumps for small tasks.

`sampling_down_factor=10`

This makes the CPU stay at a higher frequency a bit longer before scaling down.

It can improve perceived responsiveness.

`powersave_bias=200`

This adds a slight energy-saving bias while keeping performance acceptable.

For my hardware, this gives a good balance.

---

17. PP4MNK governor script

To make the CPU governor tuning easy to apply, I can use a small script.

Script: PP4MNK Ondemand Optimizer

```
#!/bin/sh
# ========================================
# PP4MNK ONDEMAND GOVERNOR OPTIMIZER
```

```
# Goal: responsive desktop without forcing max CPU all the time
# ========================================

for CPU in /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor; do
  [ -w "$CPU" ] && echo ondemand > "$CPU"
done

if [ -d /sys/devices/system/cpu/cpufreq/ondemand ]; then
  [ -w /sys/devices/system/cpu/cpufreq/ondemand/up_threshold ] &&
    echo 95 > /sys/devices/system/cpu/cpufreq/ondemand/up_threshold

  [ -w /sys/devices/system/cpu/cpufreq/ondemand/sampling_down_factor ] &&
    echo 10 > /sys/devices/system/cpu/cpufreq/ondemand/sampling_down_factor

  [ -w /sys/devices/system/cpu/cpufreq/ondemand/powersave_bias ] &&
    echo 200 > /sys/devices/system/cpu/cpufreq/ondemand/powersave_bias
fi

printf "PP4MNK ondemand governor optimization applied.\n"
```

Suggested location:

```
/root/Startup/pp4mnk-ondemand.sh
```

Make it executable:

```
chmod +x /root/Startup/pp4mnk-ondemand.sh
```

---

18. Network and browsing responsiveness

I also tested networking and browser fluidity.

The goal was not only to get high download speeds, but to improve real browsing comfort:

- faster DNS resolution,
- low latency,
- stable streaming,
- low packet loss,
- smooth page loading,
- good browser responsiveness.

Typical checks include:

```
ping -c 10 1.1.1.1
```

```
nslookup google.com 1.1.1.1
```

```
curl -w '@-' -o /dev/null -s https://www.wikipedia.org <<'EOF'
time_namelookup:  %{time_namelookup}\ntime_connect:     %{time_connect}
\ntime_starttransfer:%{time_starttransfer}\ntime_total:       %{time_total}
\nEOF
```

These tests help separate real network issues from browser or system issues.

---

19. Browser process monitoring

When a browser feels heavy, I inspect CPU and RAM usage.

Useful command:

```
ps -eo pid,comm,%cpu,%mem --sort=-%cpu | head -15
```

This quickly shows whether Opera, Firefox, Slimjet, Chromium, Xorg, JWM or another process is consuming CPU.

This is important because browser performance problems are not always caused by the operating system.

Sometimes the problem is:

- a heavy website,
- a video decoder issue,
- an extension,
- a web process,
- a GPU process,
- or a background tab.

---

20. Memory monitoring

Useful memory command:

```
free -h
```

Example target situation:

```
Mem:  11Gi total
Used: around 1.5–2.5Gi in normal use
```

```
Available: around 8–10Gi
Swap: 0B
```

This is ideal for my RAM-centric workflow.

With around 11 GiB RAM, I can safely use `/dev/shm` for browser caches without putting too much pressure on memory.

---

21. SSD write reduction strategy

The system reduces SSD writes through several layers:

21.1 Boot mode

```
qfix=copy,nosave
```

This reduces automatic save behavior.

21.2 Browser cache in RAM

Browser cache directories are redirected to:

```
/dev/shm/pp4mnk-browsers/
```

21.3 TMP in RAM

Each browser launcher sets:

```
export TMPDIR="$TMP_DIR"
```

21.4 XDG cache in RAM

Each launcher sets:

```
export XDG_CACHE_HOME="$CACHE_DIR"
```

21.5 Manual observation

I check:

```
grep Dirty /proc/meminfo
```

and browser cache directories:

```
du -sh /dev/shm/pp4mnk-browsers/*
```

---

22. Why I do not want a completely clean browser profile every boot

A completely clean profile sounds attractive, but for daily use it is not ideal.

If the profile is fully temporary, I lose:

- bookmarks,
- extensions,
- logins,
- browser settings,
- site preferences,
- history if wanted,
- productivity customizations.

That is why my final strategy is more balanced:

```
Profile: persistent
Cache: temporary in RAM
TMP: temporary in RAM
Media cache: temporary in RAM
GPU cache: temporary in RAM
```

This gives me a clean and fast browser cache without destroying my daily setup.

---

23. JWM desktop customization

I use JWM as my desktop environment because it is lightweight, fast, and suitable for EasyOS.

My visual goal is a dark, clean, developer-friendly PP4MNK style.

Main ideas:

- dark bars,
- grey/slate tones,
- minimal distractions,
- readable contrast,
- clean wallpaper,
- practical tray buttons,
- quick browser launchers,
- no unnecessary desktop clutter.

I do not use Plank in this Daedalus setup.

Instead, I prefer native JWM panels and trays.

---

24. JWM centered tray idea

A JWM tray can be aligned and adjusted with options like:

```
<Tray autohide="bottom" insert="right" width="0" height="41" layer="above"
layout="horizontal" halign="center" valign="bottom">
```

Important detail:

```
width="0"
```

In JWM, this can allow the tray to adjust closer to content size depending on the layout and included elements.

A small spacer can be added with:

```
<Spacer width="10" height="10"/>
```

This allows fine control over tray spacing.

---

25. Example JWM tray browser buttons

```
<TrayButton popup="ChatGPT" icon="/usr/share/icons/blue_chatgpt.png">exec:/
files/portable/firefox-ram.sh https://chatgpt.com</TrayButton>

<TrayButton popup="Yahoo Mail" icon="/usr/share/icons/blue_mail.png">exec:/
files/portable/firefox-ram.sh https://mail.yahoo.com/</TrayButton>

<TrayButton popup="Protopage" icon="/usr/share/icons/protopage-
icon.png">exec:/files/portable/firefox-ram.sh https://www.protopage.com/
pp4mnkt3am</TrayButton>

<TrayButton popup="Opera" icon="/usr/share/icons/opera.png">exec:/files/
portable/opera-ram.sh</TrayButton>

<TrayButton popup="Slimjet" icon="/usr/share/icons/slimjet.png">exec:/files/
portable/slimjet-ram.sh</TrayButton>
```

This keeps the desktop practical and fast.

---

26. Benchmarking approach

I created and used several benchmark-style scripts to evaluate Daedalus.

The goal is not only raw speed. The goal is to measure real usability.

Important areas:

- system load,
- memory usage,
- CPU frequency,
- browser CPU usage,
- Xorg/JWM activity,
- RAM cache behavior,
- internet latency,
- DNS speed,
- page loading time,
- dirty memory,
- SSD write behavior.

Example commands:

```
uptime
```

```
free -h
```

```
ps -eo pid,comm,%cpu,%mem --sort=-%cpu | head -15
```

```
grep Dirty /proc/meminfo
```

```
cat /proc/cmdline
```

```
cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor
```

---

27. PP4MNK Daedalus quick audit script

This script gives a fast overview of the current Daedalus state.

```
#!/bin/sh
# ========================================
# PP4MNK DAEDALUS QUICK AUDIT
```

```
# System, RAM mode, browser cache and write state
# ================================================

clear

echo "================================================="
echo " PP4MNK DAEDALUS QUICK AUDIT"
echo "================================================="
echo

echo "-------------------------------------------------"
echo "1) SYSTEM"
echo "-------------------------------------------------"
echo "Hostname      : $(hostname)"
echo "Kernel        : $(uname -r)"
echo "Architecture  : $(uname -m)"
echo "Date          : $(date)"
echo
uptime 2>/dev/null || busybox uptime

echo

echo "-------------------------------------------------"
echo "2) KERNEL CMDLINE"
echo "-------------------------------------------------"
cat /proc/cmdline

echo

echo "-------------------------------------------------"
echo "3) EASYOS STATE"
echo "-------------------------------------------------"
if [ -f /etc/rc.d/PUPSTATE ]; then
  grep -E "PUPMODE|EOSMODE|PUPSFS|SAVE|WKG|QSFS" /etc/rc.d/PUPSTATE
else
  echo "PUPSTATE not found."
fi

echo

echo "-------------------------------------------------"
echo "4) MEMORY"
echo "-------------------------------------------------"
free -h

echo

echo "-------------------------------------------------"
echo "5) DIRTY / WRITEBACK MEMORY"
echo "-------------------------------------------------"
grep -E "Dirty|Writeback" /proc/meminfo
```

```
echo

echo "------------------------------------------------"
echo "6) CPU GOVERNOR"
echo "------------------------------------------------"
cat /sys/devices/system/cpu/cpu*/cpufreq/scaling_governor 2>/dev/null | sort
| uniq -c

echo

echo "------------------------------------------------"
echo "7) TOP CPU PROCESSES"
echo "------------------------------------------------"
ps -eo pid,comm,%cpu,%mem --sort=-%cpu | head -15

echo

echo "------------------------------------------------"
echo "8) RAM BROWSER CACHES"
echo "------------------------------------------------"
if [ -d /dev/shm/pp4mnk-browsers ]; then
  du -sh /dev/shm/pp4mnk-browsers/* 2>/dev/null
else
  echo "/dev/shm/pp4mnk-browsers not found."
fi

echo

echo "------------------------------------------------"
echo "9) OVERLAY / TMPFS CHECK"
echo "------------------------------------------------"
mount | grep -E "tmpfs|overlay|aufs|union|easy_ro|easy_rw|pup_ro|pup_rw"

echo

echo "================================================"
echo " Audit complete."
echo "================================================"
```

Suggested location:

```
/root/pp4mnk-daedalus-audit.sh
```

Make executable:

```
chmod +x /root/pp4mnk-daedalus-audit.sh
```

Run:

```
/root/pp4mnk-daedalus-audit.sh
```

---

28. What I have achieved

At this stage, the system has reached a very good balance.

Boot and session

- Daedalus boots in a RAM-oriented mode.
- Automatic saving is controlled.
- The system feels clean and fast.

Browsers

- Opera, Slimjet, Chromium and Firefox can use RAM-based temporary areas.
- Browser profiles remain persistent.
- Cache writes to SSD are reduced.
- Streaming and browsing feel responsive.

Desktop

- JWM keeps the system lightweight.
- The interface can be customized with dark bars and PP4MNK-style visuals.
- Tray buttons provide fast access to web apps.

CPU

- The ondemand governor gives a good balance between speed and efficiency.
- The system avoids unnecessary full-speed CPU behavior.

SSD protection

- `/dev/shm` cache redirection reduces write pressure.
- `qfix=copy,nosave` helps prevent uncontrolled saves.
- Dirty memory remains low in normal checks.

---

29. What this setup is for

This setup is designed for:

- daily browsing,
- ChatGPT use,
- streaming,
- writing,
- web apps,

- testing browsers,
- technical experiments,
- low-write computing,
- old hardware optimization,
- and a clean RAM-centric workflow.

It is not designed to be a heavy gaming system or a modern GPU workstation.

It is designed to make older hardware feel alive, fast and useful.

---

30. Final evaluation

My conclusion is clear:

> **EasyOS Daedalus is currently the best main system for my workflow.**

It gives me the right balance of speed, simplicity, control and stability.

With RAM-based browser caches, persistent profiles, JWM customization, ondemand CPU tuning and careful write monitoring, Daedalus becomes more than a lightweight operating system.

It becomes a personal optimized workstation.

This is not a generic configuration.

It is a practical, tested, pp4mnk-style setup:

```
RAM-first
SSD-friendly
browser-focused
JWM-lightweight
old-hardware-aware
fast enough for daily work
clean enough for experimentation
stable enough to become my main system
```

That is why I am satisfied with Daedalus.

And that is why I am ready to use it as my main operating system.

---

31. Short summary

After testing several EasyOS and Puppy-based setups, I am now very satisfied with EasyOS Daedalus and I think it will become my main system.

My current setup is based on a RAM-centric workflow using `qfix=copy,nosave`, JWM, portable browsers and browser caches redirected to `/dev/shm`.

The main goal is to keep the browser profiles persistent while moving temporary cache, GPU cache, media cache and TMP files to RAM. This reduces SSD writes, improves responsiveness and keeps the system cleaner between sessions.

I also tuned the CPU governor using `ondemand`, monitor write pressure with `/proc/meminfo`, and use small audit scripts to check RAM mode, browser cache usage, process load and EasyOS state.

For my hardware and my workflow, Daedalus currently gives me the best balance between speed, stability, simplicity and control.

In short: this is my PP4MNK Daedalus main system profile — RAM-first, SSD-friendly, lightweight and very comfortable for daily use.